

ANKARA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ



Proje Adı

Hastane Randevu Sistemi Final Raporu

Ders Adı : Ağ Tabanlı Teknolojileri Ve Uygulamaları

Proje Videosu : <https://youtu.be/wg9AgMOx II>

Öğrenci Bilgileri

Ad Soyad: Bekir Köroğlu

Numara: 19291096

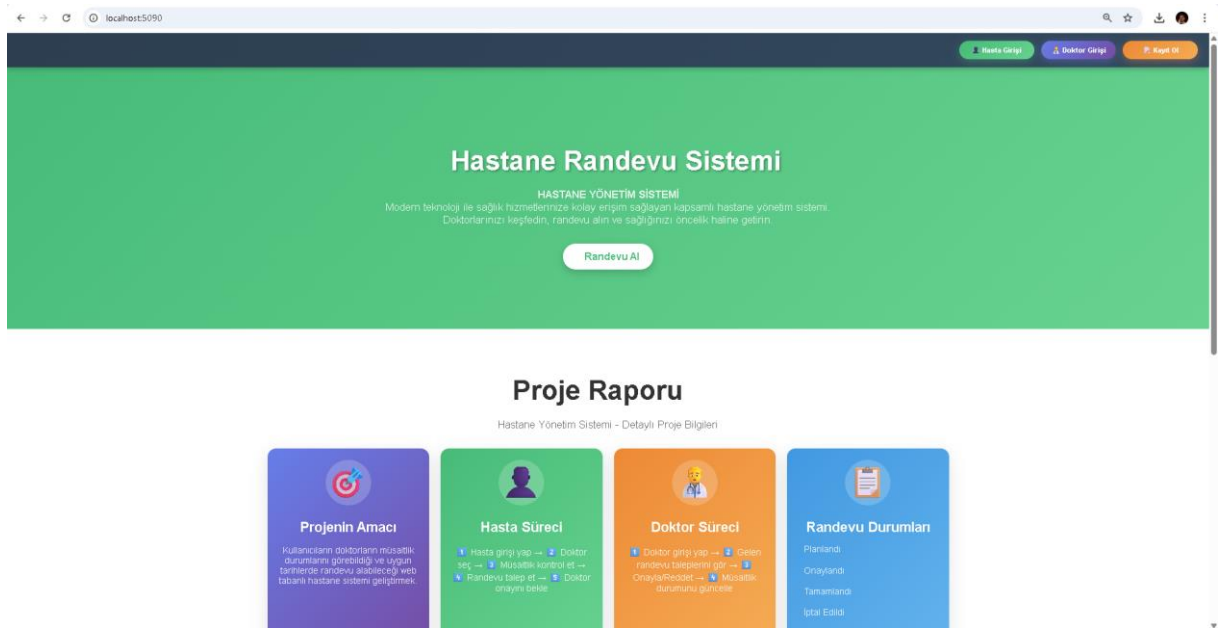
GitHub Proje Linki

<https://github.com/bekirkrgl/Ag-Tabanlı-Teknolojiler-Ve-Uygulamalari>

1. GİRİŞ

Günümüzde sağlık sektöründe dijitalleşme, hizmet kalitesinin artırılması ve süreçlerin daha düzenli yürütülmesi açısından büyük önem taşımaktadır. Özellikle hastanelerde hasta-doktor randevu süreçlerinin manuel yöntemlerle veya dağınık sistemler üzerinden yürütülmesi, zaman kaybına, iletişim problemlerine ve veri güvenliği risklerine yol açabilmektedir. Bu bağlamda, randevu yönetiminin merkezi, güvenli ve erişilebilir bir yazılım sistemi üzerinden yürütülmesi önemli bir ihtiyaç hâline gelmiştir.

Bu proje kapsamında geliştirilen **Hastane Randevu Sistemi**, hasta, doktor ve yönetici rollerini içeren web tabanlı bir uygulamadır. Sistem, kullanıcıların kimlik doğrulama işlemlerini güvenli şekilde gerçekleştirmesini, randevu süreçlerini kolayca yönetmesini ve tüm verilerin düzenli bir veritabanı yapısı içerisinde saklanmasını amaçlamaktadır. Proje, modern web teknolojileri kullanılarak geliştirilmiş olup sürdürülebilir ve genişletilebilir bir mimariye sahiptir.



2. PROJENİN AMACI VE KAPSAMI

Bu projenin temel amacı, hastanelerdeki randevu alma ve yönetme süreçlerini dijital ortama taşıyarak hem hastalar hem de sağlık personeli için daha düzenli ve kullanıcı dostu bir sistem oluşturmaktır. Geliştirilen sistem sayesinde hastalar, doktorların uygunluk durumlarını görüntüleyerek randevu alabilmekte;

doktorlar kendi randevularını takip edebilmekte; yöneticiler ise sistem genelinde kontrol ve yönetim işlemlerini gerçekleştirebilmektedir.

Proje kapsamında geliştirilen sistem aşağıdaki temel işlevleri içermektedir:

- Kullanıcı kayıt ve giriş işlemleri
- Rol tabanlı yetkilendirme (Hasta, Doktor, Admin)
- Randevu oluşturma ve görüntüleme
- Doktor uygunluk bilgilerinin yönetilmesi
- Yönetici tarafından sistem kontrolü

Bu yönüyle proje, temel bir hastane randevu sisteminden beklenen tüm işlevleri kapsamaktadır.

3. KULLANILAN TEKNOLOJİLER

Projenin geliştirilme sürecinde modern ve yaygın olarak kullanılan yazılım teknolojilerinden faydalanılmıştır. Uygulama, web tabanlı bir mimari üzerine kurulmuştur.

Projede kullanılan başlıca teknolojiler şunlardır:

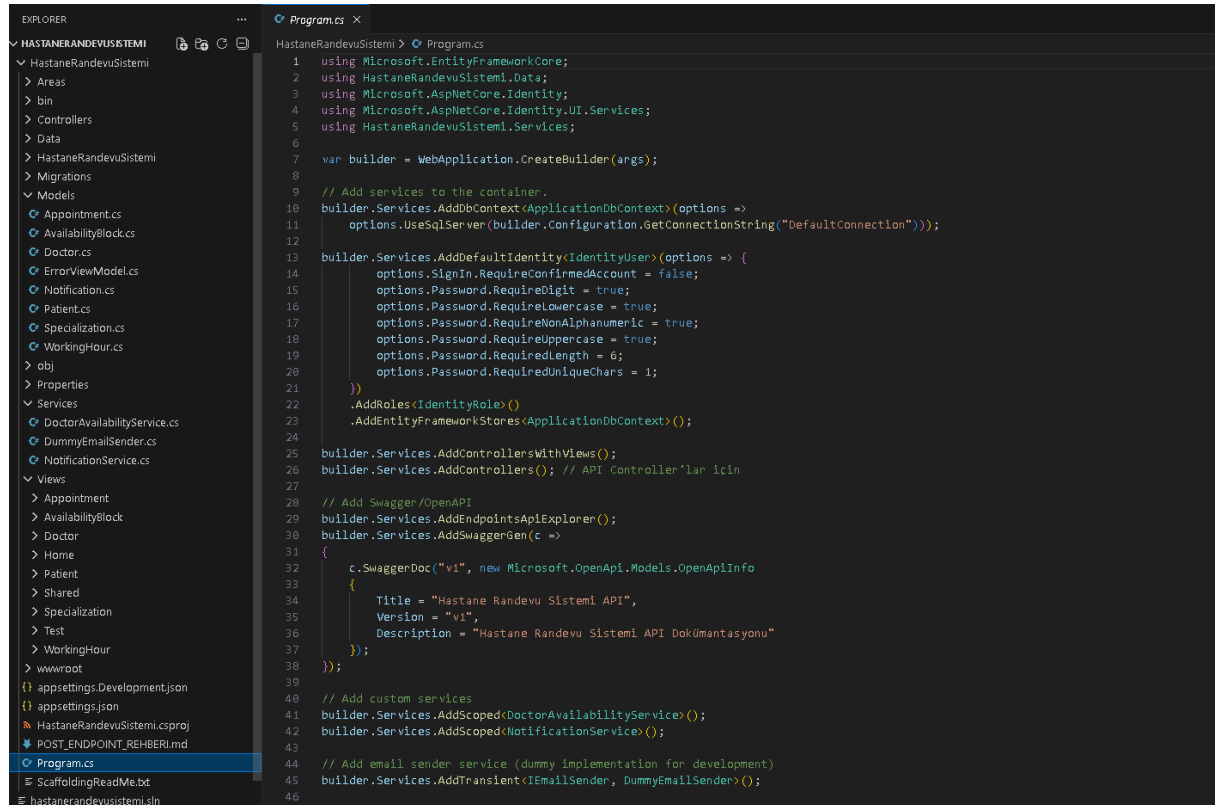
- **ASP.NET Core MVC:** Uygulamanın ana çatısını oluşturan web framework
- **Entity Framework Core:** Veritabanı işlemlerinin yönetilmesi
- **Microsoft SQL Server:** Verilerin saklandığı ilişkisel veritabanı
- **ASP.NET Identity:** Kimlik doğrulama ve yetkilendirme işlemleri
- **RESTful API:** Uygulama içi veri iletişimi
- **Swagger (OpenAPI):** API dokümantasyonu ve test ortamı

Bu teknolojiler, sistemin hem güvenli hem de sürdürülebilir olmasını sağlamıştır.

4. SİSTEM MİMARİSİ

Hastane Randevu Sistemi, **katmanlı mimari** yaklaşımına uygun olarak geliştirilmiştir. Bu mimari yapıda her katman belirli bir sorumluluğu üstlenmektedir ve katmanlar arası bağımlılık minimum seviyede tutulmuştur.

Controller katmanı, kullanıcıdan gelen istekleri karşılamakta ve gerekli yönlendirmeleri yapmaktadır. Service katmanı, uygulamanın iş mantığını yürütmekte ve Controller ile Data katmanı arasında köprü görevi görmektedir. Data katmanı ise veritabanı işlemlerinin gerçekleştirildiği katmandır. Bu yapı sayesinde kod okunabilirliği artmış, bakım ve geliştirme süreçleri kolaylaştırılmıştır. Uygulamanın giriş noktası olan **Program.cs** dosyasında; servislerin eklenmesi, kimlik doğrulama ayarları, rol tanımlamaları, middleware sıralaması ve Swagger yapılandırması gerçekleştirilmiştir.



```
1 using Microsoft.EntityFrameworkCore;
2 using HastaneRandevuSistemi.Data;
3 using Microsoft.AspNetCore.Identity;
4 using Microsoft.AspNetCore.Identity.UI.Services;
5 using HastaneRandevuSistemi.Services;
6
7 var builder = WebApplication.CreateBuilder(args);
8
9 // Add services to the container.
10 builder.Services.AddDbContext<ApplicationDbContext>(options =>
11     options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));
12
13 builder.Services.AddDefaultIdentity<IdentityUser>(options => {
14     options.SignIn.RequireConfirmedAccount = false;
15     options.Password.RequireDigit = true;
16     options.Password.RequireLowercase = true;
17     options.Password.RequireNonAlphanumeric = true;
18     options.Password.RequireUppercase = true;
19     options.Password.RequiredLength = 6;
20     options.Password.RequiredUniqueChars = 1;
21 });
22 .AddRoles<IdentityRole>()
23 .AddEntityFrameworkStores<ApplicationDbContext>();
24
25 builder.Services.AddControllersWithViews();
26 builder.Services.AddControllers(); // API Controller'lar için
27
28 // Add Swagger/OpenAPI
29 builder.Services.AddEndpointsApiExplorer();
30 builder.Services.AddSwaggerGen(c =>
31 {
32     c.SwaggerDoc("v1", new Microsoft.OpenApi.Models.OpenApiInfo
33     {
34         Title = "Hastane Randevu Sistemi API",
35         Version = "v1",
36         Description = "Hastane Randevu Sistemi API Dokümantasyonu"
37     });
38 });
39
40 // Add custom services
41 builder.Services.AddScoped<DoctorAvailabilityService>();
42 builder.Services.AddScoped<NotificationService>();
43
44 // Add email sender service (dummy implementation for development)
45 builder.Services.AddTransient<IEmailSender, DummyEmailSender>();
46
```

5. KİMLİK DOĞRULAMA VE YETKİLENDİRME

Sistemde kullanıcı doğrulama ve yetkilendirme işlemleri **ASP.NET Identity** altyapısı kullanılarak sağlanmıştır. Kullanıcılar sisteme kayıt olduktan sonra belirli şifre kurallarına uygun olarak giriş yapabilmektedir. Şifre politikaları; minimum uzunluk, büyük harf, küçük harf, sayı ve özel karakter zorunluluğu gibi güvenlik önlemlerini içermektedir.

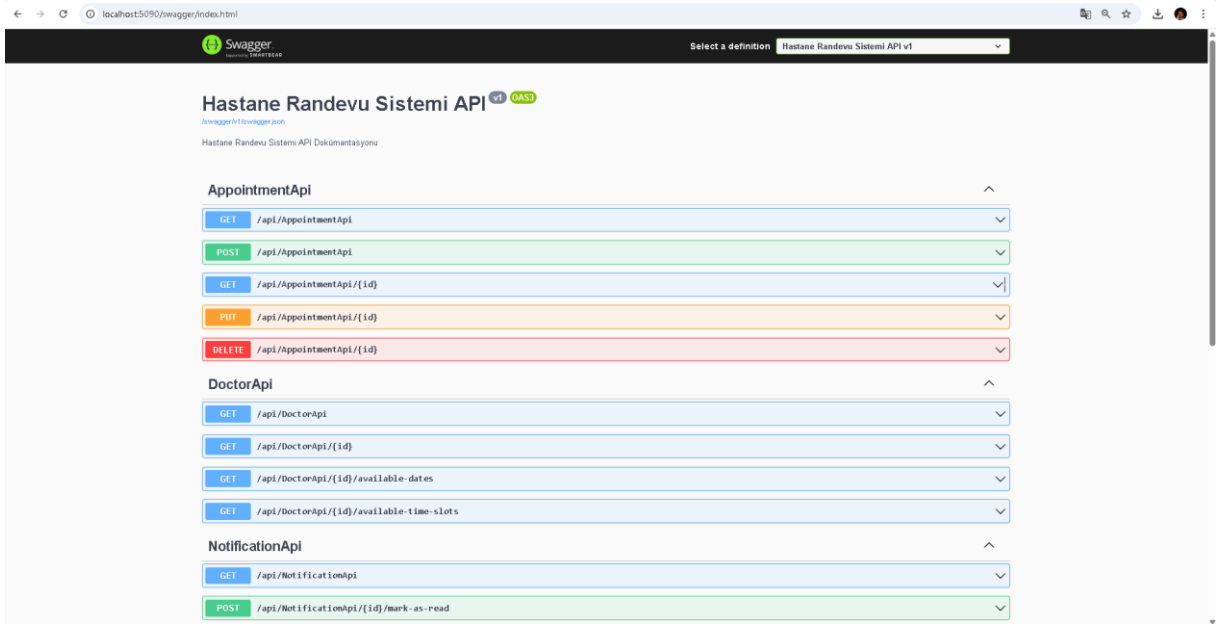
Uygulamada üç farklı rol bulunmaktadır: Hasta, Doktor ve Admin. Her rol, yalnızca kendi yetkileri dâhilinde işlem yapabilmektedir. Örneğin, hastalar randevu alabilirken yönetici yetkilerine erişemez; doktorlar ise yalnızca kendi

randevularını görüntüleyebilir. Bu yapı, sistem güvenliğini artırmakta ve yetkisiz erişimleri engellemektedir.

6. API YAPISI VE SWAGGER KULLANIMI

Uygulama içerisinde REST mimarisine uygun API controller'lar yer almaktadır. Bu API'ler, istemciden gelen istekleri alarak gerekli verileri JSON formatında döndürmektedir. API yapısı, uygulamanın modüler olmasını sağlamakta ve ileride farklı platformlarla entegrasyona olanak tanımaktadır.

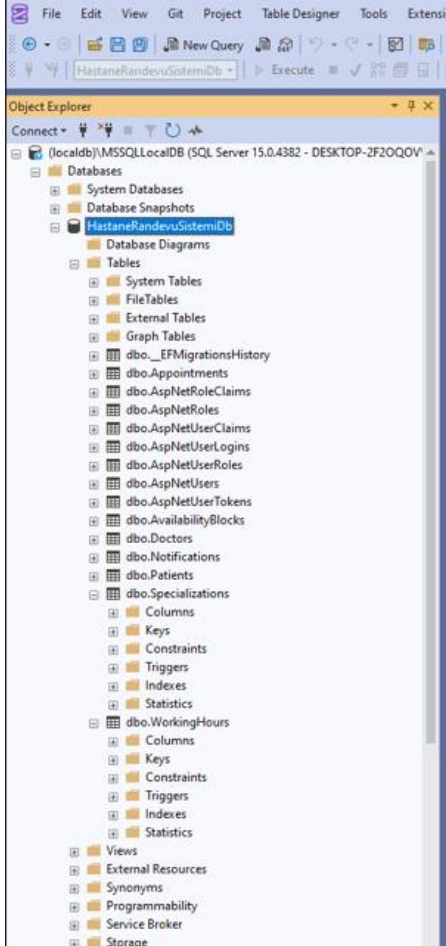
API'lerin dokümantasyonu ve test işlemleri **Swagger** arayüzü üzerinden gerçekleştirilmiştir. Swagger sayesinde tüm endpoint'ler tek bir arayüz üzerinden görüntülenebilmekte ve test edilebilmektedir. Bu durum, hem geliştirme sürecinde hem de sistemin anlaşılabilirliğinde önemli avantajlar sağlamıştır.



7. VERİTABANI TASARIMI

Veritabanı işlemleri **Entity Framework Core** kullanılarak gerçekleştirilmiştir. Kullanıcı ve rol bilgileri ASP.NET Identity tarafından oluşturulan tablolar üzerinden tutulmaktadır. Bunun yanında, randevu bilgileri ve doktor uygunluk verileri için ayrı tablolar tasarlanmıştır.

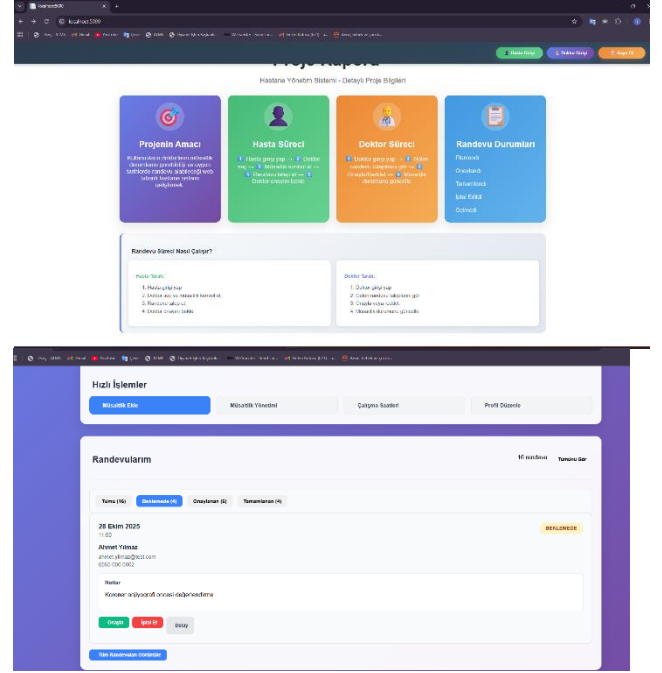
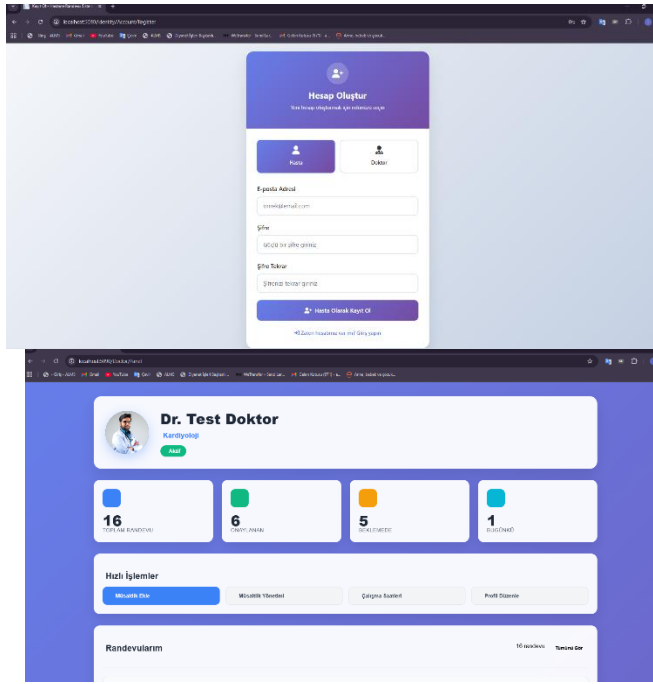
Controller katmanından gelen istekler, Service katmanına iletilmekte; burada gerekli iş mantığı uygulandıktan sonra veriler veritabanından çekilmekte veya güncellenmektedir. Bu yapı sayesinde veritabanı işlemleri doğrudan arayüz üzerinden değil, kontrollü bir katman aracılığıyla yürütülmektedir.



8. KULLANICI ARAYÜZÜ VE ÇALIŞMA MANTIĞI

Uygulamanın kullanıcı arayüzü, sade ve anlaşılır olacak şekilde tasarlanmıştır. Kullanıcıların gördüğü tüm veriler, dinamik olarak veritabanından çekilmektedir. Örneğin, hasta randevu alma ekranında görüntülenen doktor listesi, ilgili controller metodunun service katmanını çağırmasıyla veritabanındaki doktor bilgilerinden elde edilmektedir.

Benzer şekilde, bir randevu oluşturulduğunda bu işlem önce controller tarafından alınmakta, service katmanında işlenmekte ve ardından veritabanına kaydedilmektedir. Böylece kullanıcı arayüzü, kod yapısı ve veritabanı arasında doğrudan bir ilişki kurulmuş olmaktadır. Bu yapı, hocaların özellikle görmek istediği “ekran–kod–veritabanı” bağlantısını net biçimde ortaya koymaktadır.



10. SONUÇ VE DEĞERLENDİRME

Sonuç olarak geliştirilen Hastane Randevu Sistemi, proje başlangıcında belirlenen hedefleri başarıyla karşılamaktadır. Sistem; güvenli kimlik doğrulama, rol tabanlı yetkilendirme, düzenli veritabanı yapısı ve kullanıcı dostu arayüzü ile işlevsel bir çözüm sunmaktadır.

Proje, ilerleyen süreçlerde mobil uygulama entegrasyonu, bildirim sistemleri (SMS/e-posta) ve gelişmiş raporlama modülleri eklenerek daha da geliştirilebilir. Bu proje süreci, web tabanlı yazılım geliştirme, mimari tasarım ve veritabanı yönetimi konularında önemli deneyimler kazandırmıştır.